In [10]:

```
# import Pkg; Pkg.add("Interpolations")
using Plots
using CSV
using DataFrames
using Interpolations
# using ScatteredInterpolation
```

In [11]:

```
# Import battery performance data
csv_Li_ion_1 = CSV.File("Molicell_18650_2.6A - Sheet1.csv")# 10A=1C
df_Li_ion_1 = DataFrame(csv_Li_ion_1)
itp_Li_ion_1 = LinearInterpolation(df_Li_ion_1.Ahr, df_Li_ion_1.V, extrapolation_bc=Line())

csv_Li_ion_2 = CSV.File("Molicell_18650_10A - Sheet1.csv")# 20A=3.8C
df_Li_ion_2 = DataFrame(csv_Li_ion_2)
itp_Li_ion_2 = LinearInterpolation(df_Li_ion_2.Ahr, df_Li_ion_2.V, extrapolation_bc=Line())

csv_Li_ion_3 = CSV.File("Molicell_18650_20A - Sheet1.csv")# 30A=7.6C
df_Li_ion_3 = DataFrame(csv_Li_ion_3)
itp_Li_ion_3 = LinearInterpolation(df_Li_ion_3.Ahr, df_Li_ion_3.V, extrapolation_bc=Line())

csv_a123_1C = CSV.File("ANR26650M1B, 1C.csv")
df_a123_1C = DataFrame(csv_a123_1C)
itp_a123_1C = LinearInterpolation(df_a123_1C.Ahr, df_a123_1C.V, extrapolation_bc=Line())

csv_a123_6C = CSV.File("ANR26650M1B, 6C.csv")
df_a123_6C = DataFrame(csv_a123_6C)
itp_a123_6C = LinearInterpolation(df_a123_6C.Ahr, df_a123_6C.V, extrapolation_bc=Line())

# csv_a123_16C = CSV.File("A123Systems/ANR26650M1B, 16C.csv")
# df_a123_16C = DataFrame(csv_a123_16C)
# itp_a123_16C = LinearInterpolation(df_a123_16C.Ahr, df_a123_16C.V, extrapolation_bc=Line(

csv_a123_20C = CSV.File("ANR26650M1B, 20C.csv")
df_a123_20C = DataFrame(csv_a123_20C)
itp_a123_20C = LinearInterpolation(df_a123_20C.Ahr, df_a123_20C.V, extrapolation_bc=Line())

function batt_v(x, x1, x2, y1, y2)
  m = (y2-y1)/(x2-x1)
  y = m*(x-x1) + y1
end
```

Out[11]:

```
batt_v (generic function with 1 method)
```

In [12]:

```
# Diode forward voltage drop
# INPUT THE NEW DIODE
# https://www.vishay.com/docs/93804/vs-175bqg030hf4.pdf
v_drop_diode = 0.782

#INPUT THE INFORMATION FROM THE RELEVANT 2 BATTERY CHOICES


# Individual cell capacity, in [Ahr]
ahr_a123_cell = 2.56
ahr_Li_ion_cell = 2.6

# Individual cell mass, in [kg]
mass_a123_cell = .076
mass_Li_ion_cell = 0.05

# Individual cell nominal energy capacity, in [kwhr]
kwhr_a123_cell = .00825
kwhr_Li_ion_cell = .01092

# Individual cell max discharge rate
c_a123_max = 40
c_Li_ion_max = 13.4

# Number of parallel cells in pack
num_cols_a123 = 2
num_cols_Li_ion = 12

# Number of series rows in pack
num_rows_a123 = 21
num_rows_Li_ion = 14

# Pack nominal energy capacity
kwhr_a123_pack = kwhr_a123_cell * num_cols_a123 * num_rows_a123
kwhr_Li_ion_pack = kwhr_Li_ion_cell * num_cols_Li_ion * num_rows_Li_ion

# Pack amps parallel capacity
ahr_a123_pack = ahr_a123_cell * num_cols_a123
ahr_Li_ion_pack = ahr_Li_ion_cell * num_cols_Li_ion


# Pack cell mass
mass_a123_pack = mass_a123_cell * num_cols_a123 * num_rows_a123
mass_Li_ion_pack = mass_Li_ion_cell * num_cols_Li_ion * num_rows_Li_ion


pack_A_label =  string("A123 LiFePo4 (", num_rows_a123, "S", num_cols_a123, "P)")
pack_B_label =  string("Li-ion (", num_rows_Li_ion, "S, ", num_cols_Li_ion, "P,",ahr_Li_ion

println(round.([mass_a123_pack mass_Li_ion_pack], digits = 1))
println(round.([ahr_a123_pack kwhr_a123_pack;  ahr_Li_ion_pack kwhr_Li_ion_pack], digits =
```

```
[3.2 8.4]
[5.1 0.3; 31.2 1.8]
```

In [13]:

```
# Simulate across time
t = 0
delT = .05

# Power schedule
P_TAKEOFF = 16e3
P_MAX_CONTINUOUS = 12e3
P_SUSTAIN_CLIMB = 10e3
P_SUSTAIN = 6e3

# Plotting variables
t_experiment = Float64[]
I_motor = Float64[]
V_a123 = Float64[]
V_Li_ion = Float64[]
soc_a123_experiment = Float64[]
soc_Li_ion_experiment = Float64[]
c_a123_experiment = Float64[]
c_Li_ion_experiment = Float64[]
P_experiment = Float64[]
E_a123 = Float64[]
E_Li_ion = Float64[]
soc_20_Li_ion = []
soc_5_a123 = []

c_a123 = 0
c_Li_ion = 0


## Initial conditions
# Electrical power load
P_elec = P_TAKEOFF

# Start off at full charge
soc_a123 = 1
soc_Li_ion = 1

v_0_a123 = itp_a123_20C(0)
v_0_Li_ion = itp_Li_ion_1(0)

kwhr_accum_a123 = 0
kwhr_accum_Li_ion = 0

ahr_accum_a123 = 0
ahr_accum_Li_ion = 0


v_instantaneous_a123 = v_0_a123
v_instantaneous_Li_ion = v_0_Li_ion

voltage_a123_pack = v_instantaneous_a123 * num_rows_a123
voltage_Li_ion_pack = (v_instantaneous_Li_ion * num_rows_Li_ion) - v_drop_diode

if abs(voltage_Li_ion_pack - voltage_a123_pack) <= .001
  V_system = voltage_a123_pack
else
  V_system = max(voltage_a123_pack, voltage_Li_ion_pack )
end
```

```
I_a123 = P_elec / V_system
I_Li_ion = 0


# Run the simulation
for i=1:(800/delT)
  t = t + delT

  if 1==1
    # Reduce power according to battery limitations
    if c_Li_ion > c_Li_ion_max
      P_elec = P_elec - 1000
    end
  else
    # Reduce power according to schedule and/or battery limitations
    if (t > 60 || (c_Li_ion > c_Li_ion_max || c_a123 > c_a123_max)) && P_elec == P_TAKEOFF
      P_elec = P_MAX_CONTINUOUS
    end

    if (t > 180 || (c_Li_ion > c_Li_ion_max || c_a123 > c_a123_max))  && P_elec == P_MAX_CO
      P_elec = P_SUSTAIN_CLIMB
    end

    if (t > 300 || (c_Li_ion > c_Li_ion_max || c_a123 > c_a123_max))  && P_elec == P_SUSTAI
      P_elec = P_SUSTAIN
    end
  end

  # Calculate accumulated amp-hour consumption
  ahr_accum_a123 = ahr_accum_a123 + I_a123 * delT/3600
  ahr_accum_Li_ion = ahr_accum_Li_ion + I_Li_ion * delT/3600

  # Saturate AHr consumption
  if ahr_accum_a123 > ahr_a123_pack
#     ahr_accum_a123 = ahr_a123_pack
  end

  if ahr_accum_Li_ion > ahr_Li_ion_pack
#     ahr_accum_diamond = ahr_diamond_pack
  end

  # Calculate accumulated kw-hour consumption
  kwhr_accum_a123 = kwhr_accum_a123 + (I_a123 * V_system) * delT/3600/1000
  kwhr_accum_Li_ion = kwhr_accum_Li_ion + (I_Li_ion * V_system) * delT/3600/1000

  # Calculate State of Charge
  soc_a123 = 1 - ahr_accum_a123/ahr_a123_pack
  soc_Li_ion = 1 - ahr_accum_Li_ion/ahr_Li_ion_pack

  if soc_5_a123 == [] && soc_a123 < 0.05
    soc_5_a123 = t
  end

  if soc_20_Li_ion == [] && soc_Li_ion < 0.20
    soc_20_Li_ion = t
  end

  # Initialize some variables before going into the convergence loop
  isFirstLoop = true

  loopCount = 1
```

```
P_split_low = 0
P_split_high = 1
P_split = (P_split_high + P_split_low)/2

# Loop enough times to be sure to converge. Convergence is pretty decent after 15 loops,
while (loopCount < 20) || isFirstLoop == true
  I_a123 = (P_elec * (1-P_split)) / voltage_a123_pack
  I_Li_ion = (P_elec * P_split) / voltage_Li_ion_pack

  # Calculate cell discharge rate
  c_a123 = I_a123 / ahr_a123_pack
  c_Li_ion = I_Li_ion / ahr_Li_ion_pack

  # Determine cell voltages
  if c_a123 > 6
    v_instantaneous_a123_6C = itp_a123_6C(ahr_accum_a123/num_cols_a123)
    v_instantaneous_a123_20C = itp_a123_20C(ahr_accum_a123/num_cols_a123)

    v_instantaneous_a123 = batt_v(c_a123, 6, 20, v_instantaneous_a123_6C, v_instantaneous
  else
    v_instantaneous_a123_1C = itp_a123_1C(ahr_accum_a123/num_cols_a123)
    v_instantaneous_a123_6C = itp_a123_6C(ahr_accum_a123/num_cols_a123)

    v_instantaneous_a123 = batt_v(c_a123, 1, 6, v_instantaneous_a123_1C, v_instantaneous_
  end

  if c_Li_ion > 3.8
    v_instantaneous_Li_ion_2 = itp_Li_ion_2(ahr_accum_Li_ion/num_cols_Li_ion)
    v_instantaneous_Li_ion_3 = itp_Li_ion_3(ahr_accum_Li_ion/num_cols_Li_ion)
    v_instantaneous_Li_ion = batt_v(c_Li_ion, 3.8, 7.6, v_instantaneous_Li_ion_2, v_insta
  else
    v_instantaneous_Li_ion_1 = itp_Li_ion_1(ahr_accum_Li_ion/num_cols_Li_ion)
    v_instantaneous_Li_ion_2 = itp_Li_ion_2(ahr_accum_Li_ion/num_cols_Li_ion)
    v_instantaneous_Li_ion = batt_v(c_Li_ion, 1, 3.8, v_instantaneous_Li_ion_1, v_instant
  end

  # Saturate battery voltages
  if v_instantaneous_Li_ion > 4.2
    v_instantaneous_Li_ion = 4.2
  end

  if v_instantaneous_a123 > 3.4
    v_instantaneous_a123 = 3.4
  end

  # Determine pack voltages
  voltage_a123_pack = v_instantaneous_a123 * num_rows_a123
  voltage_Li_ion_pack = (v_instantaneous_Li_ion * num_rows_Li_ion) - v_drop_diode

  # Bisecting search pattern
  if voltage_Li_ion_pack > voltage_a123_pack
    P_split_low = P_split
    P_split = (P_split_high + P_split)/2
    P_split_high = P_split_high
  else
    P_split_high = P_split
    P_split_low = P_split_low
    P_split = (P_split_low + P_split)/2
  end

  # Increment loop variables.
```

```
    isFirstLoop = false
    loopCount = loopCount + 1
  end

    # Some useful console spew
    if 1==0
      println(
        t, ": ",
        " [",
        round(v_instantaneous_a123, digits=3), " ",
        round(voltage_a123_pack, digits=2), " ",
        round(c_a123, digits=1), ", ",
        round(ahr_accum_a123, digits=4), ", ",
        round(soc_a123*100, digits=1),
        "], [",
        round(v_instantaneous_Li_ion, digits=3), " ",
        round(voltage_Li_ion_pack, digits=2), ", ",
        round(c_diamond, digits=1), ", ",
        round(ahr_accum_Li_ion, digits=4), ", ",
        round(soc_Li_ion*100, digits=1),
    " ], :", round(voltage_a123_pack*I_a123 + voltage_Li_ion_pack*I_Li_ion))
  end


  append!(t_experiment, t)
  append!(c_a123_experiment, c_a123)
  append!(c_Li_ion_experiment, c_Li_ion)
  append!(soc_a123_experiment, soc_a123)
  append!(soc_Li_ion_experiment, soc_Li_ion)
  append!(V_a123, voltage_a123_pack)
  append!(V_Li_ion, voltage_Li_ion_pack)
  append!(P_experiment, P_elec)
  append!(E_a123, kwhr_accum_a123)
  append!(E_Li_ion, kwhr_accum_Li_ion)
end

[soc_5_a123, soc_20_Li_ion]
```

Out[13]:

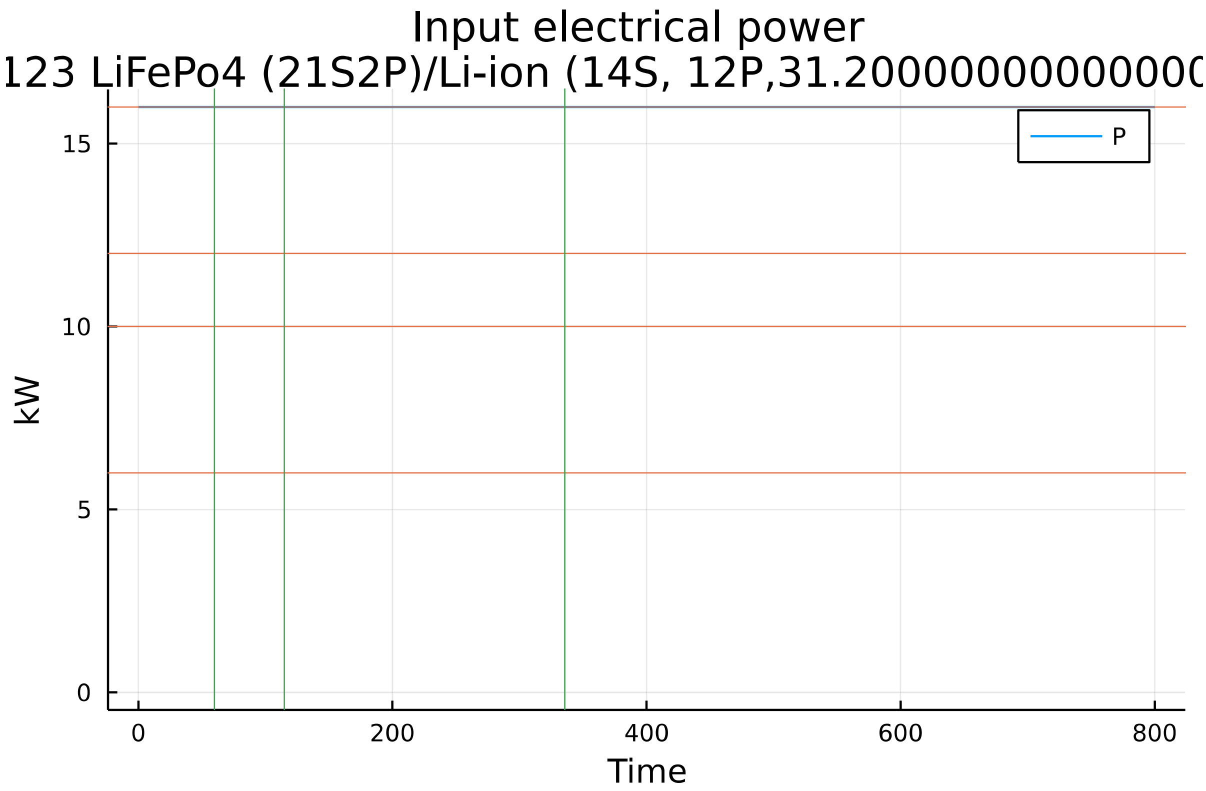
```
2-element Vector{Float64}:
 114.84999999999562
 335.65000000004204
```

In [14]:

```
plot(t_experiment, P_experiment./1000, xlabel = "Time", ylabel = "kW", ylims=[0,Inf], title

hline!([P_TAKEOFF, P_MAX_CONTINUOUS, P_SUSTAIN_CLIMB, P_SUSTAIN]./1000, lw=0.5, label="")
vline!([60, soc_5_a123, soc_20_Li_ion], lw=0.5, label="")
```

Out[14]:

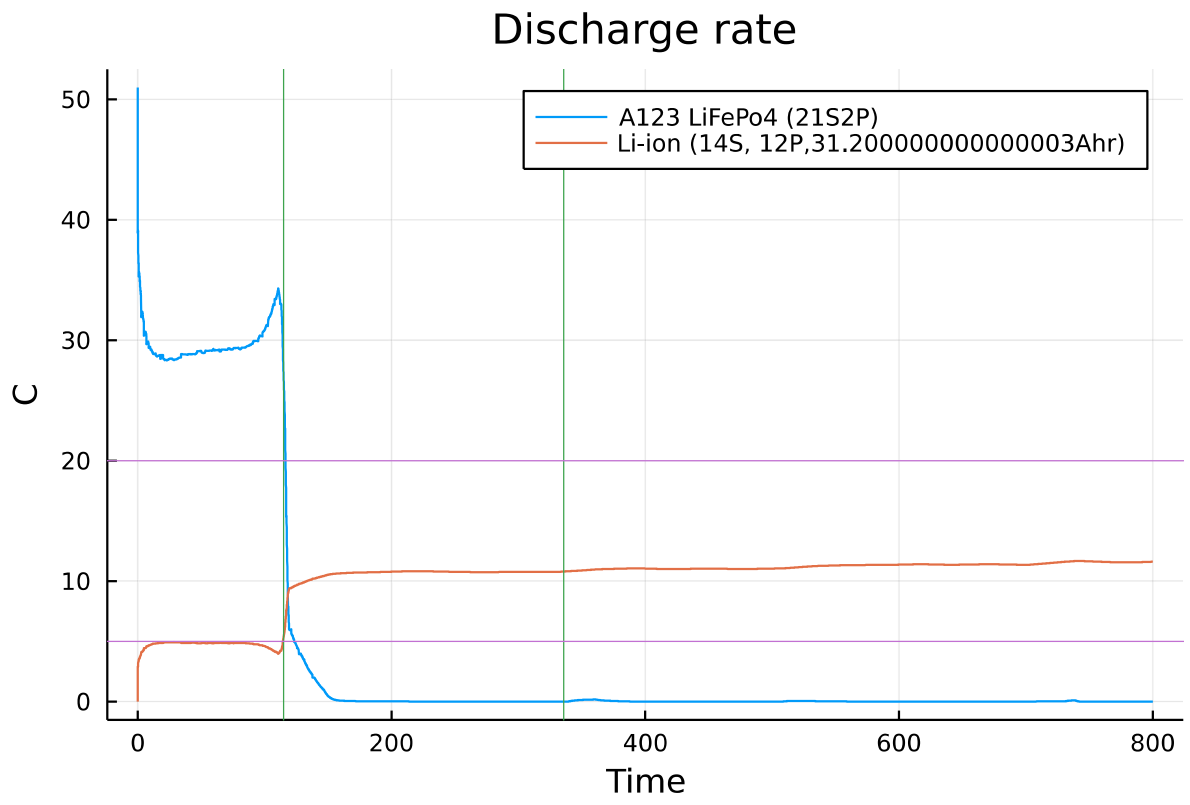


In [15]:

```
plot(t_experiment, [c_a123_experiment c_Li_ion_experiment], title="Discharge rate", xlabel

vline!([soc_5_a123, soc_20_Li_ion], lw=0.5, label="")
hline!([5,20], label="", lw=0.5) # Max nominal discharge rates
# savefig("plot.png")
```

Out[15]:



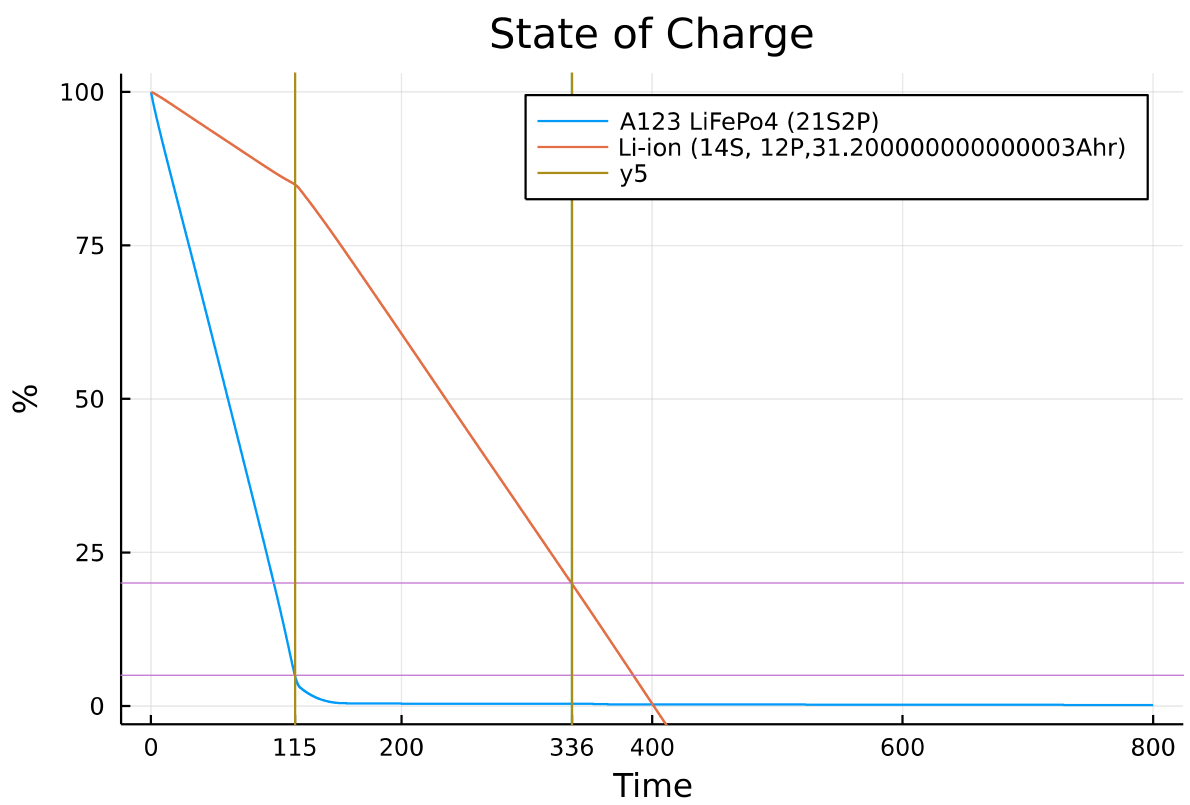
In [16]:

```
b = plot(t_experiment, [soc_a123_experiment soc_Li_ion_experiment].*100, title="State of Ch

vline!([soc_5_a123, soc_20_Li_ion], lw=0.5, label="")
hline!([5,20], label="", lw=0.5)  # SoC limits

old_xticks = xticks(b[1]) # grab xticks of the 1st subplot
new_xticks = (round.([soc_5_a123, soc_20_Li_ion]), string.(Int.(round.([soc_5_a123, soc_20_
vline!(new_xticks[1])
keep_indices = findall(x -> all(x .≠ new_xticks[1]), old_xticks[1])
merged_xticks = (old_xticks[1][keep_indices] ∪ new_xticks[1], old_xticks[2][keep_indices] ∪
xticks!(merged_xticks)
```

Out[16]:

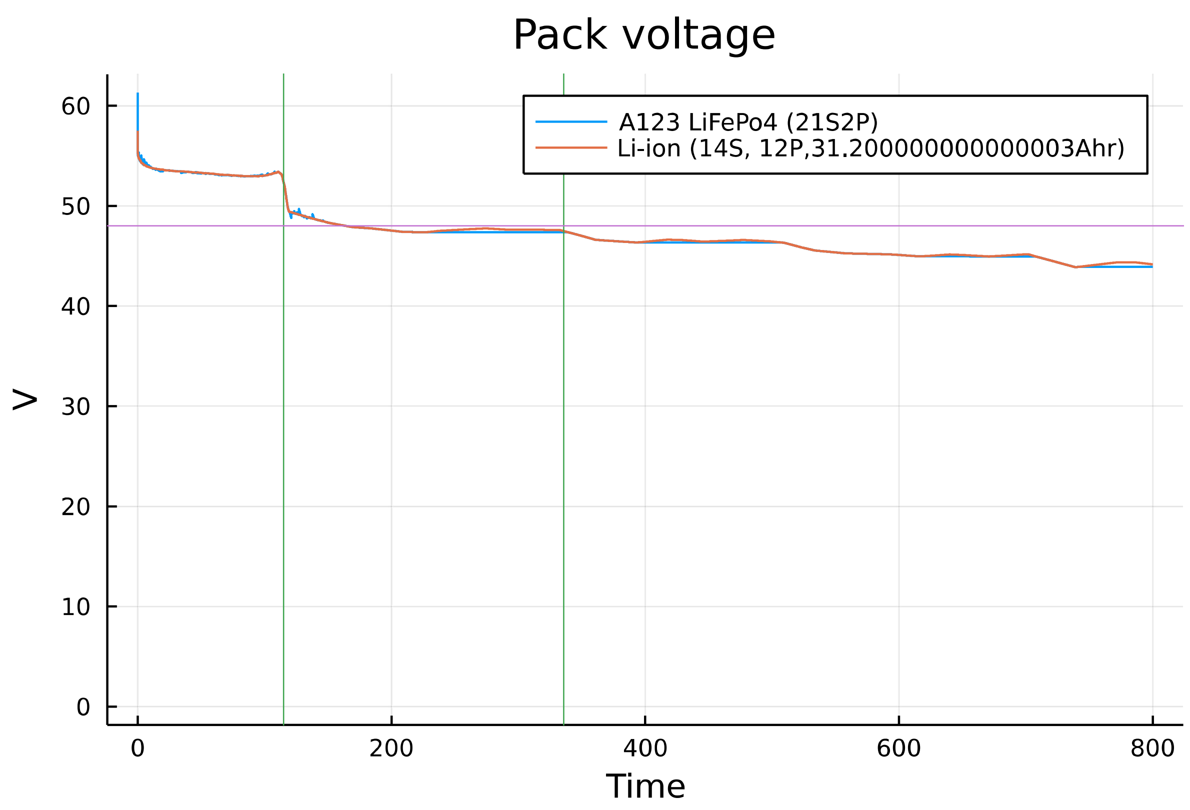


In [17]:

```
plot(t_experiment, [V_a123 V_Li_ion], xlabel = "Time", title="Pack voltage", ylabel = "V",

vline!([soc_5_a123, soc_20_Li_ion], lw=0.5, label="")
hline!([48], label="", lw=0.5)
```

Out[17]:

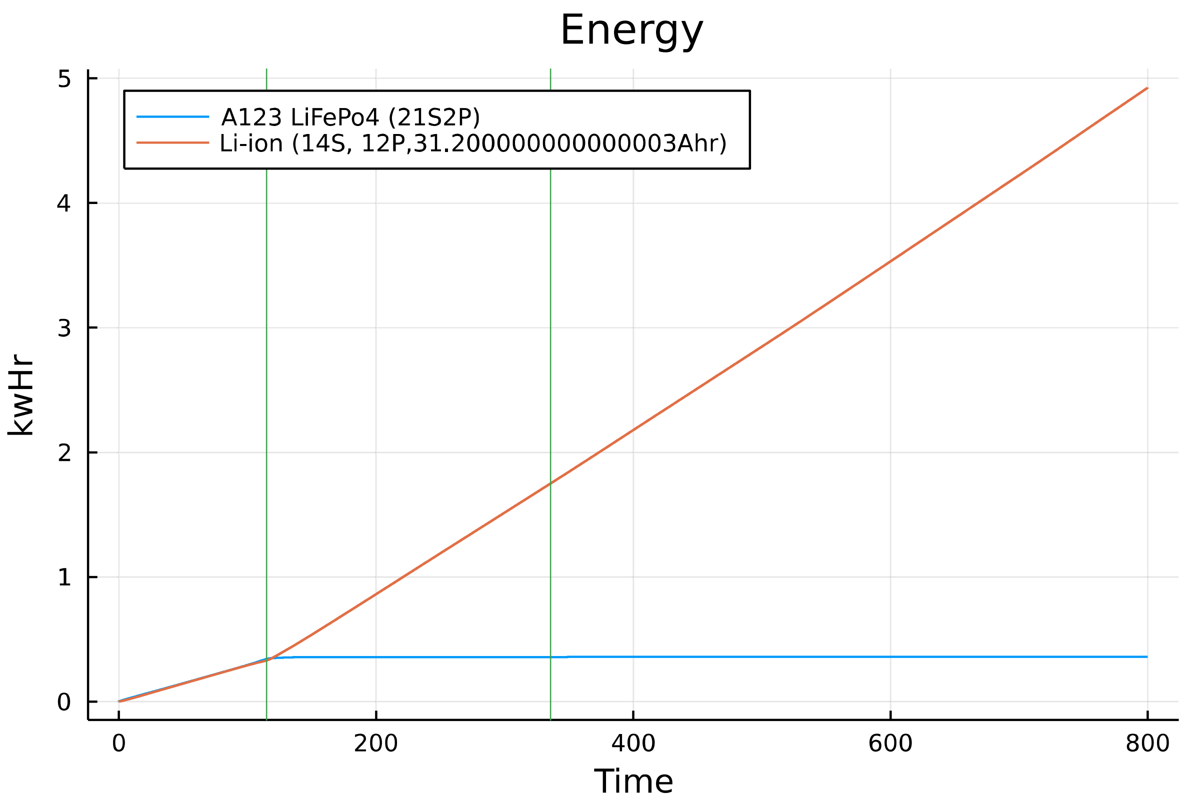


In [18]:

```
plot(t_experiment, [E_a123 E_Li_ion], title="Energy", xlabel = "Time", ylabel = "kwHr", lab

vline!([soc_5_a123, soc_20_Li_ion], lw=0.5, label="")
# hline!([0.6,3], label="Total energy", lw=0.5)444444556
```

Out[18]:



In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: